Enabling the Generation, Preservation & Use of Records and Memories of Everyday Life

Khai N. Truong and Gregory D. Abowd
College of Computing & GVU Center
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280 USA
+1 404 545 1036
{khai, abowd}@cc.gatech.edu

ABSTRACT
Our daily lives provide us with a great deal of records and memories that we want to access at some point in the future. As a result, we often spend our time and effort trying to preserve much of these experiences. To assist in the process, automated capture and access applications attempt to relieve humans of the burden of manually documenting these experiences. To allow application developers to build such applications more easily, we have developed INCA, an infrastructure for capture and access applications. In this paper, we show how this infrastructure can facilitate the construction of these applications, and demonstrate its use through the development of three sample applications.

Keywords
Ubiquitous computing, capture and access applications, infrastructure, application development.

INTRODUCTION
People often take pictures to capture a moment, write notes to document an experience, and keep paperwork as records of important interactions. These are all examples from everyday lives of how people capture information for some later use. As humans, our manual capture procedures are often far from ideal. What is the best picture to take to preserve the memory of a moment? What notes should be written to summarize an important point during a lecture? Unfortunately, many of us are not very good at creating accurate records of an experience; as a result, these records are often biased, incomplete, and in some cases may even contain errors.

One of the general themes in ubiquitous computing is the construction of devices and applications to support the automated capture of live experiences and the future access of those records. Over the years, many systems have been built to capture and access experiences in classrooms [1, 2, 8, 16], offices [17], meetings [6, 18, 20, 23, 26-28], conferences [9], and other generalized experiences [22, 25]. There is much variety in the types of captured information —including audio, video, electronic ink, handwritten notes, and prepared presentations. Access of these captured experiences typically occurs in the near term (on the order of seconds to minutes later [13, 14]) or long term (but usually no longer than a year later). Capture and access applications can exist on a single self-contained device (such as the Audio Notebook [22] or DynaBook [28]) but most use a variety of capture devices (such as Classroom 2000 [1, 4]) and assume that access occurs on a different access application (such as NotePals [8]).

Despite this variety of research, we argue in this paper that there is still a wide range of capture and access applications to be explored and evaluated. Furthermore, it is still difficult to build these applications —meaning that those who might have the most creative applications for automated capture and access are prohibited from realizing their visions. We have developed an infrastructure to facilitate in the construction of capture and access applications. This infrastructure, INCA, aids application developers by:

1. Lowering the barrier for building capture and access applications,
2. Making complex applications easier to evolve and to fine-tune to meet the users' needs, and
3. Enabling the construction of a larger variety of applications previously unexplored.

Overview of Paper
This paper presents an infrastructure aimed at simplifying the process of building capture and access applications. We begin by describing three sample capture and access applications that motivate the need for a wider variety of this class of ubiquitous computing applications. We explain how these applications fit in with respect to existing applications and how they differ from related work. We outline the requirements and issues involved in the development of capture and access applications and describe how they inform the design of an infrastructure to support the application developers. The three capture

and access applications are revisited to demonstrate how the infrastructure facilitates their development. We conclude with a discussion of related work and our plans for future work.

EXPLORING THE DESIGN SPACE OF CAPTURE & ACCESS APPLICATIONS

To motivate the variety of capture and access applications, we briefly describe three sample applications we have built. We will motivate the problem that each is attempting to solve and present a simple solution that we were able to build using INCA. The rest of the paper then describes how the INCA framework supports the design and implementation of this class of system, revisiting features of these sample applications to demonstrate how more complex features are supported.

The sample applications are:

- **What Were We Talking About?** A near-term audio recording system designed to assist with the resumption of interrupted conversations.
- **The Family Video Archive.** A context-based capture and retrieval system for home video.
- **WebMemex.** An instrumented Web browser that facilitates the recording of an individual's browsing activity with context-based recommendations.

These three applications explore different aspects of the various dimensions in the design space of capture and access applications described by Truong *et al.* [24] What Were We Talking About? and The Family Video Archive demonstrate extreme timescales of capture and access; the former is an example of a near-term application, while the latter is a long-term application. The three applications capture different kinds of data: audio, video, and web pages. Additionally these applications represent the types that can be found in the existing literature (What Were We Talking About?) and those previously unexplored (The Family Video Archive and WebMemex). WebMemex demonstrates how an existing application, in this case a commercial Web browser, can have capture and access features added to it.

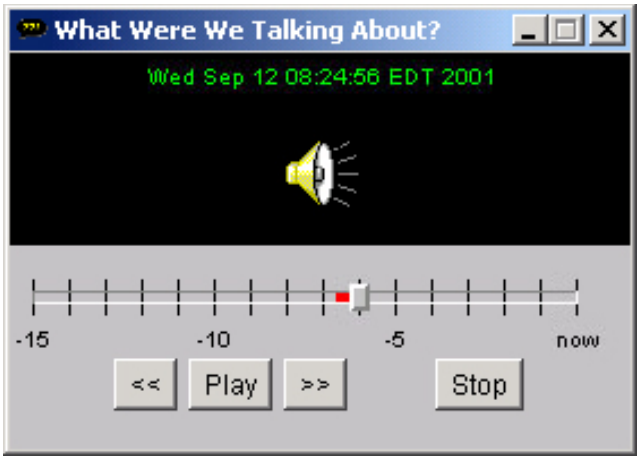

Figure 1. Interface to a simple conversation reminder tool.

What Were We Talking About?

Motivation

Have you ever been in the middle of a conversation that has drifted in some direction and forgotten what the main point of discussion was? When you notice that the conversation has strayed off the main topic, or you simply want to return to a previous topic, you often ask the question, "What were we talking about?" But sometimes it is difficult to remember the gist of a conversation, even if it occurred only minutes ago. This application is intended to support a quick way to revisit recent conversations in ways similar to the Where Were We system [14].

How It Works

In the simplest case, we used a single, mobile device that travels with an individual to continuously record audio that would then be available for playback to the user in a manner similar to a tape recorder. We implemented this service on a laptop computer, but one could easily envision this running on a general purpose handheld device or even a custom designed information appliance built specifically for this service of audio reminding. Unlike a tape recorder, however, this service continues to capture audio even when playback of previously recorded information is accessed.

The audio portion of a conversation is recorded and the user can quickly access any part of the recent past. After some time (we chose 15 minutes), the captured audio is discarded, meaning this system only supports the access to near-term recorded conversations. The access interface (see Figure 1) provides a timeline that allows the user to jump back and skim the recent past, relistening to a conversation in order to return to a previous topic of discussion. The right end of the timeline represents the current time. When the user wants to play back some portion of the conversation, she can move the scrub handle on the timeline backwards and playback will commence immediately. The portion of the conversation that has been played back is highlighted in red on the timeline. The user can quickly jump forward or backward along the timeline by dragging the scrub or by nudging it slightly forward (using >>) or backward (using <<). Pressing the stop button cancels playback.

The Family Video Archive

Motivation

Many of us capture home videos of our family as a memory of some special occasion. These videos accumulate over time and it is difficult to catalogue them and revisit desired scenes from the past. As digital video becomes more pervasive, opportunities arise for supporting more flexible means of capturing, cataloguing and reviewing family videos [12]. The functionality we envisioned was that of a searchable video archive that

could serve up video sequences that matched queries of the sort, "Show me all of the videos from the 1960's with my Grandma in it," or "Let me see all of the important events in my life."

How It Works

We have prototyped a family video album service to support tagging of video scenes with information describing the scene (e.g., who is in it, where it is, what event is recorded). As a demonstration, we are exploring tagging already recorded video archives, but we have designed the system with the intent to support the simultaneous recording and tagging of recorded video. This application supports a simple form of digital video organization and editing; that is, the location and assembly of relevant video sequences from a large repository.

Figure 2. Interface for the attaching known properties to recorded video segments.

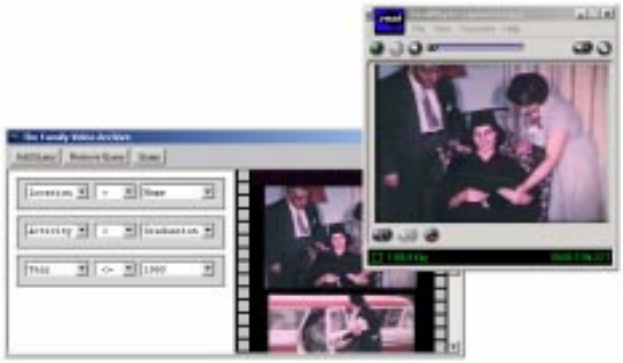

Figure 3. Interface for specifying a family video to dynamically generate for viewing.

The capture application (see Figure 2) is instrumented to attach known properties (or attributes such as time and location information) to segments of the captured video. The video is stored in an archive along with all its metadata information. When the user wishes to watch some family videos, she specifies a query of what she wants to view (see Figure 3). As the user queries for the video she wants, different clips that match the query show up in the video reel —essentially generating a dynamic family video. The user can then begin viewing this video, or select the video clip or group of videos to play.

WebMemex

Motivation

The Web is what many of us rely on for information and entertainment. As we experience the Web, we visit many different sites, and view many things. However, when we want to retrieve previously viewed information, often we have either forgotten to bookmark the relevant URL or we cannot use the history mechanism stored locally on the browser machine. Histories do not persist and require the user to remember features of the Web page, such as its URL or title, that may be difficult to remember. Often, it is some portion of the content of a page or other context describing the user's visit to the page (e.g., when the user was in a particular place or at a particular time or after viewing some other page) that are cues that we remember.

In his 1945 Atlantic Monthly article, Vannevar Bush described his vision of the *memex*, a generalized capture and access application [5]. He noted that a "record … must be continuously extended, it must be stored, and above all it must be consulted." WebMemex is built in the same spirit of *memex*, but geared towards records of URLs that have been visited in the past. This application demonstrates being able to mark associations between captured artifacts (through keywords).

How It Works

A common Web browser (Internet Explorer®) was augmented with a simple capture capability that serves as a new history feature. When a web page is visited its contents are captured and stored in a repository accessible to the user at a later time. The Web page is analyzed and tagged with keywords extracted from the HTML as it is stored.

This annotated Web history can be used to support a number of access features, and we demonstrate here a near-term capability. As a user browses new Web pages, the access application (shown as a pop-up window in Figure 4) suggests related URLs that the user has visited from the past. By taking the keywords from the current web page the user is viewing, the access application can retrieve previous URLs that she visited in the past with matching keywords as well. The related URLs are shown in small window outside of the Web browser. After a short period of visibility, the list of related URL is

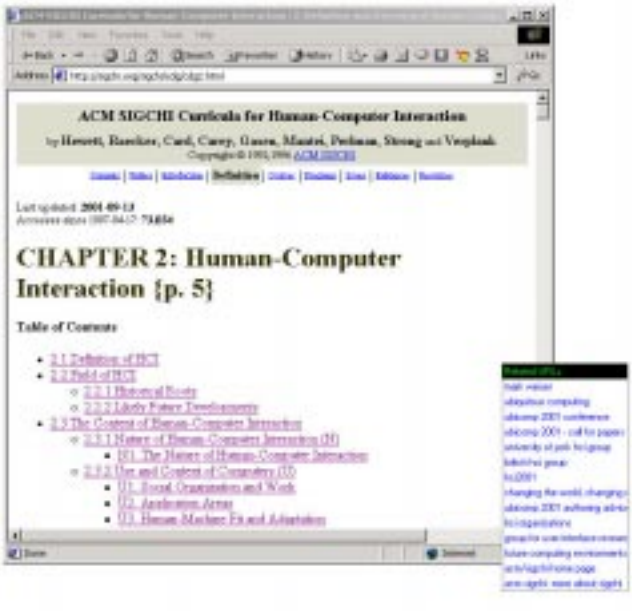

Figure 4. Web browser augmented to suggest related URLs from a user's personal Web surfing history.

collapsed to a small window that can be expanded again when the user wishes to consult it.

This application provides suggestions of related material in the same manner as the Remembrance Agent [19]. However, we limit this application to work only with Web pages encountered from a user's own personal web viewing experience. As a user continues to use WebMemex, the list of Web pages is continually extended with each Web page visited. All URLs are stored and the stored records are consulted for a list of related Web page.

SUPPORTING APPLICATION DEVELOPMENT

Essence and Accident in Developing Capture and Access Applications

The point of every capture and access application is to preserve some portion an experience so that it can be retrieved and used in the future. Broadly speaking, a capture and access application needs to:

- preserve relevant details from an experience,
- mark associations between a variety of captured artifacts, and
- provide interfaces to review past experiences in a time-efficient manner.

Truong *et al.* have previously defined five dimensions of the capture and access application space that help designers address these general user requirements [24]:

1. ***Who*** are the users during capture and access? It is important to identify how many users are involved in the separate capture and access phases, what roles they have in this application domain, and whether captured records are public, private or a mixture.
2. ***What*** is captured and accessed? An experience is defined in terms of some set of artifacts that are manipulated and information streams that are generated. Which of these artifacts and streams are important to review later on, and what level of fidelity is required between the live experience and the playback of the captured experience?
3. ***When*** does capture and access occur? How often does the captured experience occur and is there any pattern that predicts when it occurs? How much time lag is there between the live experience and the expected time of access?
4. ***Where*** does capture and access occur? Does capture or access occur in a well-defined location or set of locations? Is mobility during capture or access important?
5. ***How*** is capture and access performed? What devices and tools are required during the live experience and must be instrumented in order to facilitate capture of activity? What devices are used to access a past experience?

From a software engineering perspective, the above requirements and dimensions address *essential* tasks in the design of automated capture and access applications [3]. To facilitate the exploration of this application space, designers need to focus their skills on domain-specific answers to these requirements and dimensions.

However, the creation of a capture and access application involves many *accidental* tasks —activities that are necessary to create a working software system but are not central to the problem domain of the application itself— that can overwhelm the development effort. Capture and access applications are typically a confederation of components that must work together seamlessly. As a result, application developers spend a lot of time creating the glue that allows these independent, distributed, and heterogeneous systems to work together.

When the burden of addressing accidental tasks becomes too great for a class of applications, then the variety of applications developed decreases. Such is the case for automated capture applications. Our solution to this problem is embodied in a software framework, the INCA (Infrastructure for Capture and Access) toolkit, which we describe next. INCA provides a small set of key abstractions to overcome many of the accidental tasks of creating capture and access applications.

THE INCA TOOLKIT

Using two key software engineering principles, separation of concerns and abstraction, it is possible to make the building of capture and access applications easier.

Separation of concerns

Complex software systems can be divided into parts that perform specific functions, and the isolation of functionality makes it easier to design, implement and evolve a given system.

To apply the practice of separation of concerns, we look for architectural similarities common across all capture and access applications. We find that across these applications:

- Part of the system is responsible for the **capture** of information.
- Part of the system is responsible for the **storage** of the information.
- When information needs to be converted into different formats and types, part of the system must **transduce** the information.
- When multiple streams of information are captured, part of the system must **integrate** related pieces of information.
- Part of the system is responsible for the **access** of the information.

INCA provides an interface and an encapsulated module to aid in the capture, storage, transduction and access of the information. Thus, the part of the system responsible for the capture of information implements the *Capturer* interface and instantiates a *CaptureModule* object. Similarly, there are *Storer*, *Transducer*, and *Accessor* interfaces and *StorageModule*, *TransductionModule*, and *AccessModule* object classes for those respective portions of the system. We wrap support for the integration of information directly into support for the access of the information, such that when information is requested, related streams of information are jointly provided.

Abstraction

Implementation details for common services can be hidden from an application programmer.

- While a few systems operate on a single stand-alone device, most systems are built using multiple networked devices.
- There is much variety in the kinds of data (or information) involved in different applications.

Rather than needing to worry about issues pertaining to data distribution, what IP address devices are running on, and network protocols, INCA provides developers with a network abstraction. This abstraction allows application developers to create applications to function without needing to know how the different parts communicate with one another. A *Registry* object maintains a list of the available modules that handle the capture, storage, transduction and access of information. This object completely encapsulates all communication between relevant components. As a result, the developer can write code for complex distributed systems that involves multiple devices in the same manner as she would for an application that runs on a single device, without any knowledge of the inner workings.

Abstracting away from the details of the information being captured allows the infrastructure to generically provide support to a variety of applications designed for different domains. By viewing captured data as only raw bytes with tagged attributes, the infrastructure is able to handle any number of different kinds of data in the same generic fashion.

BUILDING APPLICATIONS WITH INCA

Earlier in the paper, we motivated three sample capture and access applications:

- What Were We Talking About?
- The Family Video Archive
- WebMemex

We described how these applications worked without detailing how they were constructed. In this section, we will now explain how INCA facilitates the construction and evolution of these applications.

What Were We Talking About?

Building It With INCA

As a short-term audio reminder system, there is no need for this capture and access application to use any kind of permanent storage for the captured information. To make the information quickly available, the capture application, *WaveCapturer*, captures audio in 4-second chunks. As audio data is captured, the *WaveCapturer* application uses an instance of a *CaptureModule* object to make that data available to the access portion of the system running on the same device. The access application (see Figure 1) uses an *AccessModule* to subscribe to the captured media and caches the information until the user wants to listen, at which time it begins smooth playback of all the captured audio chunks from when the user wants to listen.

Extending The Application

There are scenarios when it makes sense to involve more than one device in this short-term capture and access of a conversation or meeting. In some situations, the room might be better suited to do the capture, such as when the room is instrumented to capture a large meeting better than a personal device.

Because we have separated the capture and access of the audio into two separate applications, we can easily extend the system we have built to handle this distributed situation. For every room for which we want to provide this functionality, we can install just the capture application in that location. The *WaveCapturer* application is slightly modified so that location information is tagged to each audio chunk as it is captured. The access application is also modified to accept input of the user's location history (whether it is sensed implicitly or explicitly entered by the user) so it can subscribe to the audio captured wherever the user has

been in the past 15 minutes. From the user's perspective, the application will function in the same manner as it did on a single capture and access device.

From the developers' perspective, the construction of this application only requires them to focus on the specific (essential) requirements they were trying to support — being able to record audio and being able to playback audio, as well as providing a simple and intuitive user interface. Only subtle changes are needed to evolve the system from the basic application that runs on a single device to a distributed and multi-device version.

The Family Video Archive

Building It With INCA

The Family Video Archive is a long-term capture and access system that allows access to captured video that has persisted over a long period of time. Video sequences related by a variety of content or context cues, such as time ("all the videos from 1953-1960"), place ("at church"), or people present ("of dad"), are easily collected through a simple query interface.

The *VideoTagger* application (see Figure 2) takes previously captured home video and provides users with a way of attaching meta-information to the video (arbitrarily segmented into 4 minute sequences). Because it is a long term capture and access application, information must persist. INCA provides a *Repository* object class which implements a permanent storage for captured information in a placeless manner similar to that of the Presto system [10] and supports an attribute-based retrieval method. This object gives application developers a generic storage facility without any additional programming effort. For this application, the *Repository* object class is extended (or subclassed) into a *VideoRepository* class. Because video data is typically very large, the *VideoRepository* stores pointers to video chunks rather than storing the chunks in the database. After video is captured and stored over the years, the Family Video Archive access application (see Figure 3) provides users with an interface to specify queries for a video they want dynamically generated for viewing — where the video generated is one pieced together from the different parts previously recorded. Hence, once the user inputs a query, the application uses an instance of the *AccessModule* to request the video clips specified. This request will propagate to the *VideoRepository* where it searches through its set of captured data for information that matches the query.

Extending The Application

As mentioned previously, this application was prototyped to use already recorded videos that have been created over the past 30+ years. As the application is described above, users specify the attributes that are tagged to the previously recorded video clips. We have designed this system such that it can also support the simultaneous recording and tagging of video. With emerging sensing technology, it is possible to get contextual information through more implicit means, such as:

- Where the video footage is taken,
- Who is in it, and
- When it is being recorded.

With the capture and metadata tagging of the video separated from the content-based access of the video, evolution is eased. While we built the *VideoTagger* application to capture video both in the live setting as well as to handle already previously captured video, it really does not affect the access application.

WebMemex

Building It With INCA

WebMemex is a single application that both captures the user's Web surfing history and also accesses URLs that have been visited in the past that are related to the current Web page. Because it uses historical information, the generic *Repository* storage provided by INCA is used to store the captured Web pages. This application uses COM objects to control the Internet Explorer® Web browser. As the user visits different Web sites, the user's current URL is tagged with a list of keywords summarizing that Web page and then captured by an instance of a *CaptureModule* object. An instance of the *AccessModule* object is used to request URLs previously captured and also tagged with keywords that match those on the list of keywords for the current Web page. Thus, as a user is surfing the Web, she is provided with additional pieces of information: Web pages that are relevant to the current page (see Figure 4).

Extending The Application

While this application was intentionally built to demonstrate the ability to add capture and access services to a common application (in this instance, Internet Explorer®), ultimately this same capability could be provided through a centralized Web proxy service. Essentially the same kind of functionalities as the basic application we have built can be supported by augmenting a Web proxy to:

- capture the URL the user wishes to visit and tags it with the keywords for that page,
- request previously visited URLs with matching keywords with the current URL, and
- return the page the user wants to visit with some JavaScript generated and appended to show the list of related Web pages that the user has viewed in the past.

In doing so, all other Web browsers, not just Internet Explorer, can provide users with suggestions of related URLs from their past Web surfing history. In addition, tagging the URLs with details of the time of capture and location of the client browser can facilitate longer-term historical searches by the user.

The development of this proxy is simplified by taking the code that was previously written to augment the Internet Explorer® application and plugging it into a Web proxy. The only major change the developer must make pertains to the presentation of the history window shown in the bottom right of Figure 4. In this version of the WebMemex, JavaScript is used to handle the creation of a window that displays the list of related URLs.

RELATED WORK: EXISTING FRAMEWORKS

There has been previous work that looks at some of the challenges involved in providing infrastructural support for more general exploration of the potential uses of capture and access applications. The STREAMS work introduced a technique for capturing multiple streams of information as separate, single medium streams that can be temporally integrated [7]. The Tivoli work at Xerox PARC explored how to coordinate multiple applications to work collectively to capture multiple streams of information [15]. Lifestreams uses time to coordinate the display of all the documents people create [11]. Presto, on the other hand, describes a placeless method for storing information that allows related pieces of information to be grouped together and retrieved [10]. Multimedia documents demonstrate how SMIL (and the use of time) can coordinate the playback of the captured experience [21]. This previous body of work presents key features that are desired from a single infrastructure for capture and access applications.

The work we have presented in this paper provides all the functionalities mentioned here —integrating these features into a single framework that manages all the implementation details involved. Beyond this goal, our work allows designers to decompose the development process into smaller parts that are easier to attack separately, and enables the development of reusable and customizable building blocks for further investigation. In doing so, we present a community of application developers with the opportunity to explore the construction of applications in a variety of domains —to build applications that explore the full range of possibilities in the capture and access design space.

CONCLUSIONS AND FUTURE WORK

We introduced INCA, an infrastructure level support for the development of capture and access applications. We described INCA's two layers of abstraction —network abstraction and information abstraction. The infrastructure also supports the separation of concerns pertaining to the capture, storage, transduction, integration and access of information. Combined together, these features of INCA can simplify the implementation of applications. We demonstrated its use through the development of three sample applications. We showed that INCA enables the construction of novel capture and access applications, the building an existing application found in the literature and the retrofitting of a common application with capture and access capabilities. We also presented how this work supports the ability to extend and evolve each of these applications —an important feature in the building and fine-tuning of an application to meet the users' true needs. To further validate INCA as an effective tool for building capture and access applications, in the future, we will look to show that it supports the construction of applications that vary on these design dimensions for capture and access applications.

Our research is aimed at providing application developers with an infrastructure that makes it easier to build capture and access systems. We have completed the first version of the capture and access infrastructure and have made this distribution publicly available. We have provided the infrastructure to members of our research group as well as several other research institutes. We will investigate the infrastructure's ease-of-adoption by programmers. Feedback from the developers will be elicited to understand the kinds of applications that were actually built and the difficulties that were encountered.

ACKNOWLEDGMENTS

The authors are members of the Future Computing Environments (FCE) group at Georgia Tech. We would like to thank Jason A. Brotherton, Maria de Graca Pimentel, and Anind K. Dey for helping to shape the direction of this project. Much of this work was funded through various organizations (including the National Science Foundation, DARPA, and the Army Research Lab) and a number of industrial sponsors, including the members of the Aware Home Research Initiative, Sun Microsystems and Hewlett-Packard.

REFERENCES

1. Abowd, G.D., *Classroom 2000: An experiment with the instrumentation of a living educational environment.* IBM Systems Journal, 1999. **38**(4). pp. 508-530.
2. Bacher, C., *et al.* Authoring on the Fly. A New Way of Integrating Telepresentation and Courseware Production. In *Proceedings of ICCE 1997.* (Kuching, Sarawak, Malaysia), 1997.
3. Brooks, F.P., *No silver bullet: Essence and accidents of software engineering.* IEEE Computer, 1987. **20**(4). pp. 10-19.
4. Brotherton, J.A., Bhalodia, J.R., and Abowd, G.D. Automated Capture, Integration, and Visualization of Multiple Media Streams. In *Proceedings of IEEE Multimedia and Computing Systems.* (Austin, TX), 1998.
5. Bush, V., *As We May Think*, in *Atlantic Monthly*. 1945.
6. Chiu, P., *et al.* NoteLook: Taking Notes in Meetings with Digital Video and Ink. In *Proceedings of ACM Multimedia 1999.* (October 30-November 5, Orlando, FL), 1999, pp. 149-158.

7. Cruz, G. and Hill, R. Capturing and Playing Multimedia Events with STREAMS. In *Proceedings of ACM Multimedia 1994.* (San Francisco, CA.), 1994.
8. Davis, R.C., *et al.* NotePals: Lightweight Note Sharing by the Group, for the Group. In *Proceedings of CHI 1999*. (May 15-20, Pittsburgh, PA), 1999, pp. 338-345.
9. Dey, A.K., *et al.* The Conference Assistant: Combining Context-Awareness with Wearable Computing. In *Proceedings of ISWC 1999.* (October 20-21, San Francisco, CA), 1999, pp. 21-28.
10. Dourish, P., *et al.* Using Properties for Uniform Interaction in the Presto Document System. In *Proceedings of UIST 1999.* (November 7-10, Asheville, NC), 1999.
11. Freeman, E.T. *The Lifestreams Software Architecture*. Ph.D. Thesis, Department of Computer Science, Yale University, 1997.
12. Girgensohn, A., *et al.* A Semi-Automatic Approach to Home Video Editing. In *Proceedings of UIST 2000*. (November 5-8, 2000, San Diego, CA), 2000, pp. 81-89.
13. Hindus, D. and Schmandt, C. Ubiquitous Audio: Capturing Spontaneous Collaboration. In *Proceedings of Computer Supported Collaborative Work 1992*. (October 31-November 4, Toronto, Canada), 1992, pp. 210-217.
14. Minneman, S. and Harrison, S. Where were we: Making and using near-synchronous, pre-narrative video. In *Proceedings of ACM Multimedia 1993.* 1993, pp. 1-6.
15. Minneman, S., *et al.* A confederation of tools for capturing and accessing collaborative activity. In *Proceedings of ACM Multimedia 1995.* (November 5-9, San Francisco, CA), 1995, pp. 523-534.
16. Mukhopadhyay, S. and Smith, B. Passive Capture and Structuring of Lectures. In *Proceedings of ACM Multimedia 1999.* (October 30-November 5, Orlando, FL), 1999, pp. 477-487.
17. Mynatt, E.D., *et al.* Flatland: New Dimensions in Office Whiteboards. In *Proceedings of CHI 1999*. (May 15-20, Pittsburgh, PA), 1999, pp. 346-353.
18. Pedersen, E.R., *et al.* Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. In *Proceedings of ACM INTERCHI 1993.* (April 24-29, Amsterdam, The Netherlands), 1993, pp. 391-398.
19. Rhodes, B.J. *Just-In-Time Information Retrieval*. Ph.D. Thesis, Media Laboratory, MIT, 2000.
20. Richter, H., *et al.* Integrating Meeting Capture within a Collaborative Team Environment. In *Proceedings of UbiComp 2001*. (September 31-October 2, Atlanta, GA), 2001.
21. Shirmohammadi, S., Ding, L., and Georganas, N., *An Approach for Recording Multimedia Collaborative Sessions: Design and Implementation.* Journal of Multimedia Tools and Applications, 2001.
22. Stifelman, L.J. *The Audio Notebook.* Ph.D. Thesis, Media Laboratory, MIT, 1997.
23. Streitz, N.A., *et al.* DOLPHIN: Integrated Meeting Support across Liveboards, Local and Remote Desktop Environments. In *Proceedings of Computer Supported Collaborative Work 1994.* (October 22-26, Chapel Hill, NC), 1994, pp. 345-357.
24. Truong, K.N., Abowd, G.D., and Brotherton, J.A. Who, What, When, Where, How: Design Issues of Capture & Access Applications. In *Proceedings of UBICOMP 2001*. (September 31-October 2, Atlanta, GA), 2001.
25. Wactlar, H., *et al. Informedia Experience-on-Demand, Capturing, Integrating and Communication Experience across People, Time and Space.*

    http://www.informedia.cs.cmu.edu/eod.
26. Weber, K. and Poon, A. Marquee: A tool for real-time video logging. In *Proceedings of CHI 1994.* (April 24-28, Boston, MA), 1994, pp. 58-64.
27. Whittaker, S., Hyland, P., and Wiley, M. Filochat: Handwritten notes provide access to recorded conversations. In *Proceedings of CHI 1994.* (April 22-28, Boston, MA), 1994, pp. 271-277.
28. Wilcox, L., Schilit, B.N., and Sawhney, N. Dynomite: A Dynamically Organized Ink and Audio Notebook. In *Proceedings of CHI 1997.* (March 22-27, Atlanta, GA), 1997, pp. 186-193.